Bachelor's thesis

Information and Communications Technology

2025

Eevi Leväniemi

Natural Language Processing for Automating Queries in Business Intelligence Systems

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2025 | 66 pages

Eevi Leväniemi

Natural Language Processing for Automating Queries in Business Intelligence Systems

Natural Language Processing (NLP) offers the potential to simplify interactions with Business Intelligence (BI) systems. Non-technical users often face challenges when querying data as BI tools typically require proficiency in query languages such as SQL, limiting their usability. NLP technologies can make BI systems more accessible by automating SQL query generation.

This thesis examined the integration of NLP models into BI systems to translate natural language queries into SQL. Advanced pretrained models, including T5, BERT, and GPT, were fine-tuned, and a prototype was developed using PostgreSQL, FastAPI, and Power BI. The focus was on designing a user-friendly solution rather than achieving enterprise-level scalability.

The findings showed that NLP models, particularly T5 and BERT, can effectively generate SQL queries. The integration demonstrated the potential of NLP-driven BI systems to improve accessibility by providing a visually appealing and intuitive interface. While challenges in scalability and handling complex query structures remain, the research highlighted opportunities for further development, including refining model accuracy through active learning, incorporating domain-specific adaptations, leveraging user feedback, and addressing ethical considerations.

Keywords:

Natural Language Processing, Business Intelligence, SQL Query Automation, NLP Models, Data Accessibility, BI Systems

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2025 | 66 sivua

Eevi Leväniemi

Luonnollisen kielen käsittely kyselyiden automatisoinnissa liiketoimintatietojärjestelmissä

Luonnollisen kielen käsittelyllä (Natural Language Processing, NLP) on mahdollista yksinkertaistaa vuorovaikutusta liiketoimintatiedon (Business Intelligence, BI) järjestelmien kanssa. Ei-tekniset käyttäjät kohtaavat usein haasteita tiedon kyselyissä, sillä BI-työkalut vaativat yleensä kyselykielten, kuten SQL:n, osaamista, mikä rajoittaa niiden käytettävyyttä. NLP-teknologiat voivat tehdä BI-järjestelmistä saavutettavampia automatisoimalla SQL-kyselyitä.

Tässä opinnäytetyössä tutkittiin NLP-mallien integrointia BI-järjestelmiin luonnollisen kielen kyselyiden kääntämiseksi SQL:ksi. Kehittyneitä, esikoulutettuja malleja, kuten T5, BERT ja GPT, hienosäädettiin, ja prototyyppi kehitettiin käyttäen PostgreSQL:ää, FastAPI:ta ja Power BI:tä. Ratkaisun suunnittelussa painotettiin käytettävyyttä yritystason skaalautuvuuden sijaan.

Tulokset osoittivat, että NLP-malleista erityisesti T5 ja BERT voivat tehokkaasti generoida SQL-kyselyitä. Integraatio korosti NLP-pohjaisten BI-järjestelmien potentiaalia parantaa saavutettavuutta luomalla visuaalisesti miellyttävän ja helppokäyttöisen käyttöliittymän. Vaikka skaalautuvuuteen ja monimutkaisiin kyselyrakenteisiin liittyviä haasteita on edelleen ratkaisematta, tutkimus toi esiin jatkokehitysmahdollisuuksia, kuten käyttäjäpalautteen hyödyntämisen ja eettisten näkökulmien huomioimisen.

Asiasanat:

Luonnollisen kielen käsittely, liiketoimintatieto, SQL-kyselyiden automatisointi, NLP-mallit, tiedon saavutettavuus, BI-järjestelmät

Contents

Pictures

Tables

List of abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BI	Business Intelligence
EDA	Exploratory Data Analysis
GPT	Generative Pre-trained Transformer
NLP	Natural Language Processing
SQL	Structured Query Language
SSBI	Self-Service Business Intelligence
T5	Text-To-Text Transfer Transformer

1 Introduction

Data has become the backbone of modern organizations, driving decisions at all levels. Business Intelligence (BI) systems facilitate this process by offering tools to analyze, visualize, and interpret complex datasets (Chen et al., 2021). However, these systems often pose accessibility challenges for non-technical users who may lack expertise in query languages such as SQL or the ability to navigate complex dashboards. This dependency on technical teams creates inefficiencies and delays in decision-making, underscoring the need for more accessible BI solutions (Nambiar & Mundra, 2022).

Natural Language Processing (NLP) presents an innovative approach to overcoming these challenges by allowing users to interact with BI systems through natural language queries. Such systems have the potential to democratize data access, making advanced analytics available to a broader range of users (Gupta et al., 2020). Despite these promises, existing NLP-driven BI solutions struggle with handling complex queries and often lack robust integration into practical business workflows (Zhang & Liu, 2021). Research in this domain frequently focuses on the theoretical capabilities of NLP models, leaving their real-world applications in BI contexts underexplored.

This thesis explores the potential of fine-tuned NLP models, including T5, BERT, and GPT, to automate SQL query generation and improve the usability of BI systems for non-technical users. It investigates how effectively these models can enhance business intelligence workflows by addressing challenges in query generation, evaluating their integration, practical applications, and limitations.

To systematically explore these goals, Chapter 2 presents a literature review analyzing the intersection of NLP and BI, discussing existing BI tools, NLP advancements, and their integration into business intelligence. Chapter 3 describes the methodology, outlining the fine-tuning and evaluation of NLP models, as well as their implementation within a BI system. Chapter 4 details the implementation and results, covering the process of building the final NLP-

driven BI solution, the empirical evaluation of model performance, and an overview of the system's ability to translate natural language queries into SQL and deliver actionable insights through Power BI. Finally, Chapter 5 discusses the implications of these findings, considering business applications, limitations, and future research directions.

By bridging theoretical advancements in NLP with real-world applications, this thesis aims to contribute to the development of more accessible and inclusive data analytics tools. It seeks to explore how NLP-driven BI solutions can improve user accessibility and decision-making, laying the groundwork for future improvements in intelligent, user-friendly BI interfaces.

2 Literature Review

This literature review examines the intersection of Business Intelligence (BI) and Natural Language Processing (NLP).

2.1 Business Intelligence (BI) Systems

Business Intelligence (BI) refers to the methodologies and technologies that organizations employ for the analysis and management of business-related data. BI systems facilitate the collection, integration, analysis, and visualization of business information, enabling organizations to make well-informed decisions based on trustworthy and timely insights (Tavera Romero et al. 2021).

2.1.1 Definition and Evolution of BI Systems

Business Intelligence (BI) systems serve as mechanisms that convert unprocessed data into valuable insights, thereby facilitating informed decision-making. The relevance of BI has intensified, particularly in the light of the emergence of Industry 4.0, which employs technologies such as Big Data, IoT, and AI to optimize decision-making frameworks (Tavera Romero et al. 2021). These systems are pivotal for organizations, enabling them to swiftly adapt to changing market conditions and refine their strategic choices (Paradza & Daramola 2021). Furthermore, BI's role in enhancing organizational agility has become increasingly significant, allowing firms to respond proactively to market dynamics (Siau & Chen 2020).

The integration of data warehouses and data lakes within BI systems enhances the ability to process vast amounts of information, ensuring that organizations can derive insights from both structured and unstructured data sources (Nambiar & Mundra 2022). Additionally, BI systems can significantly benefit from leveraging the efficiency optimization strategies commonly employed in large-scale NLP models. Techniques such as mixed precision training and

parallel computing can improve the overall performance of BI systems, particularly when dealing with extensive datasets and complex analyses (Mei et al. 2024). Furthermore, the combination of BI and Big Data analytics improves the capacity to handle various data sources and provide actionable insights, thus positioning organizations to gain a competitive edge (Adewusi et al. 2024).

2.1.2 BI Tools and Technologies

Conventional BI tools, such as SQL-based dashboards, demand significant technical skill, limiting their accessibility for users without technical expertise (Tavera Romero et al. 2021). Newer platforms, including Power BI and Tableau, enhance data visualization but continue to face challenges with the complexity of query generation (Al-Okaily et al. 2023). Self-Service BI (SSBI) aims to make BI more accessible by allowing users to independently derive insights, though gaps in intuitive querying remain (Al-Okaily et al. 2023). Tools like Power BI and Tableau simplify data access and enable report creation without advanced technical skills, thus supporting improved decision-making (Adewusi et al. 2024).

The advent of foundation models allows NLP to further enhance SSBI tools by enabling more natural and complex query interactions. Large pre-trained models, requiring minimal customization, can interpret diverse user intents, significantly enhancing both accuracy and user experience (Paaß & Giesselbach 2023). Transformer models employing self-supervised learning techniques improve generalization across tasks, benefiting BI systems by enhancing their responsiveness to varied user queries (Kotei & Thirunavukarasu 2023). Additionally, effective BI implementation requires a data-driven organizational culture (Tavera Romero et al. 2021) and a flexible IT infrastructure to support advanced BI tools and facilitate rapid data integration (Chen & Siau 2020).

2.1.3 Data Sources for BI Systems

BI systems typically utilize data warehouses or data lakes to store vast amounts of structured and unstructured data. These repositories are essential for supporting advanced analytics by enabling organizations to analyze real-time data streams alongside historical data, thereby enhancing decision-making processes (Nambiar & Mundra 2022). By facilitating advanced analytics and reporting, data warehouses and data lakes allow organizations to gain insights from diverse data sources, which strengthens their overall business intelligence capabilities (Tavera Romero et al. 2021).

2.1.4 Role of BI in Decision-Making

The influence of BI on decision-making spans sectors such as finance, healthcare, and retail by delivering real-time insights that inform strategic actions, offering a competitive advantage for organizations able to respond swiftly to market changes (Paradza & Daramola 2021; Tavera Romero et al. 2021). Enhancements like knowledge distillation and model pruning further optimize NLP-powered BI systems by reducing model size and increasing inference speed, crucial for real-time data analysis (Mei et al. 2024). In the Big Data era, advanced analytical tools transform raw information into actionable intelligence, helping organizations manage complex datasets effectively (Adewusi et al. 2024). Additionally, NLP capabilities in BI systems can automate responses to customer inquiries, streamlining data processing and analysis (Just 2024). Tailoring BI applications to specific user needs also enables better decision-making by aligning insights and tools with the appropriate stakeholders within the organization (Passlick et al. 2023).

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field of artificial intelligence focused on enabling machines to comprehend, interpret, and interact with human language.

Utilizing diverse techniques and algorithms, NLP systems can process and analyze vast amounts of natural language data, facilitating meaningful communication between humans and machines (Just 2024).

2.2.1 Overview of NLP Models

Notable models in the NLP domain, such as GPT, BERT, and T5, have demonstrated significant effectiveness in processing natural language queries and converting them into structured formats like SQL, essential for BI applications (Al-Okaily, Teoh & Al-Okaily 2023). Each model brings unique strengths to NLP tasks: GPT excels in generating coherent text with its autoregressive design, BERT enhances contextual understanding through its bidirectional architecture, and T5 offers a versatile text-to-text approach suitable for various tasks, impacting their utility in BI systems (Wang et al. 2023).

Foundation models trained on extensive datasets amplify performance in BI-related NLP tasks, offering adaptability with minimal domain-specific data for SQL query generation and understanding business entities (Paaß & Giesselbach 2023). Recent self-supervised learning advancements in transformer-based models improve training efficiency, enabling better comprehension of linguistic nuances and enhancing features such as Named Entity Recognition (NER), which is crucial for identifying essential entities like customer names or product IDs in BI queries (Kotei & Thirunavukarasu 2023; Wang et al. 2023).

To optimize these models for BI, methods like quantization and model compression reduce computational demands while preserving performance (Mei et al. 2024). The evolution of pre-trained models has transformed NLP capabilities, enabling targeted fine-tuning for enhanced performance across applications (Wang et al. 2023). Ongoing advances in NLP also strengthen BI systems' data analysis capacities and facilitate customer service automation, expanding BI systems' engagement options (Just 2024).

Advanced NLP techniques are key to increasing BI systems' accessibility, allowing natural language interactions that meet diverse user needs (Passlick et al. 2023). However, challenges remain, particularly in handling ambiguity and real-time query processing. Self-supervised learning can further bolster NLP model robustness, supporting generalized application across different contexts (Kotei & Thirunavukarasu 2023). Given natural language's inherent ambiguity, accurately interpreting user intent is a continuous challenge (Wang et al. 2023).

2.3 Natural Language Processing in Business Intelligence

Natural Language Processing (NLP) is increasingly being explored in the context of Business Intelligence (BI) to facilitate user interaction with data.

2.3.1 Benefits of NLP Integration in BI

Incorporating Natural Language Processing (NLP) into Business Intelligence (BI) systems offers significant advantages, transforming how organizations engage with data. A primary benefit is the enhanced accessibility for non-technical users, as NLP removes the need for SQL expertise, enabling individuals from various technical backgrounds to interact seamlessly with BI systems (Al-Okaily, Teoh & Al-Okaily 2023). This democratization of data access accelerates decision-making, allowing users to query information in natural language and simplifying engagement with complex datasets (Tavera Romero et al. 2021).

Enhanced user satisfaction is another substantial benefit of NLP integration, as it provides a straightforward, intuitive approach to retrieving insights. Improving the BI user experience through NLP can positively impact the overall system efficacy, which is often measured by user satisfaction. Ensuring high data quality and usability aligns closely with the goal of integrating NLP into BI systems (Paradza & Daramola 2021; Al-Okaily, Teoh & Al-Okaily 2023). Additionally, the combination of NLP with a flexible IT infrastructure supports

adaptability and operational agility, enabling BI systems to meet the dynamic requirements of organizations (Siau & Chen 2020).

2.3.2 Advancements in NLP Techniques

Recent advancements in model training techniques, such as knowledge distillation, facilitate the transfer of knowledge from larger models to more lightweight versions, enhancing efficiency while maintaining high performance and scalability in BI applications (Mei et al. 2024). Integrating BI with Big Data analytics provides organizations with a holistic view of market trends and consumer behavior, underscoring the critical role of NLP in elevating BI systems (Adewusi et al. 2024).

Self-supervised learning techniques have also contributed to the development of robust NLP models that effectively comprehend user queries, significantly improving NLP's role in BI (Kotei & Thirunavukarasu 2023). Techniques such as sentiment analysis, named entity recognition (NER), and chatbots further enhance customer engagement, demonstrating the impact of pre-trained language models (PTMs) on the accuracy and efficiency of query generation. PTMs enable BI systems to translate natural language into structured data queries, promoting ease of use and insight generation (Wang et al. 2023). Foundation models like BERT and GPT establish a strong framework for diverse NLP tasks in BI, enhancing user experience and supporting data-driven decision-making (Paaß & Giesselbach 2023).

2.3.3 Challenges and Limitations of NLP in BI

While current NLP-driven BI systems, such as Power BI's Q&A feature, allow users to ask questions in natural language, they often fall short in handling complex queries, especially where nuanced or ambiguous language is involved, which can impact the accuracy of insights (Wang et al. 2023; Just 2024). Enhancing training techniques and model fine-tuning is crucial to making these

systems more adaptable, allowing for better comprehension of user intent and delivery of precise responses (Kotei & Thirunavukarasu 2023). Tailoring NLP systems to specific user needs through query categorization and application scenario taxonomy further bolsters BI system effectiveness (Passlick et al. 2023).

Implementing NLP in BI systems, however, poses challenges, such as scalability limitations, query accuracy, and the computational costs of running complex algorithms. Techniques like model pruning and knowledge distillation help mitigate these costs, ensuring efficient integration of large foundation models into BI applications (Paaß & Giesselbach 2023). Automating responses to common inquiries enhances operational efficiency, enabling staff to focus on complex, human-centered tasks. Additionally, NLP applications in knowledge management streamline the identification of market opportunities and consumer needs, enriching the BI ecosystem (Just 2024).

2.4 User Interaction in Business Intelligence

User interaction within Business Intelligence (BI) systems is crucial for maximizing the usability and impact of data analytics. As BI tools advance, their design and functionality must support non-technical users who may lack the skills to navigate complex interfaces or use SQL, often posing a barrier to effective data analysis (Tavera Romero et al. 2021). Enhancing user interaction within BI systems allows individuals to access insights seamlessly and make informed decisions. Natural Language Processing (NLP) integration represents a significant advancement by enabling users to pose questions in natural language, bypassing traditional query methods and making BI more accessible to non-technical users (Paradza & Daramola 2021).

The rise of Self-Service BI (SSBI) has further empowered users by allowing them to independently generate insights without specialized skills, as demonstrated by platforms like Power BI and Tableau, which prioritize user-friendly data visualization and exploration (Al-Okaily, Teoh, & Al-Okaily 2023).

However, existing NLP capabilities in these tools still encounter challenges in handling complex queries or accurately interpreting natural language nuances (Just 2024). Foundation models such as BERT and GPT have shown substantial improvements in the accuracy and flexibility of BI interactions, enabling more effective responses to diverse user queries (Paaß & Giesselbach 2023).

Incorporating feedback loops is essential for organizations aiming to refine BI systems continuously. Actively gathering user feedback and implementing iterative adjustments ensures that BI tools evolve in line with user needs and preferences (Kotei & Thirunavukarasu 2023). Classifying interaction scenarios and analyzing feedback systematically can provide insights into user preferences and challenges, leading to targeted design improvements (Passlick et al. 2023).

User feedback plays an instrumental role in shaping BI systems' effectiveness, allowing for ongoing enhancements in usability and satisfaction (Adewusi et al. 2024). Understanding user interaction is key to ensuring that BI systems effectively meet diverse needs, making them an essential part of data-driven decision-making.

2.5 Future Directions

Future advancements will require high data and model training quality to ensure successful implementation of NLP in BI systems. Comprehensive datasets and rigorous fine-tuning processes will play a pivotal role, alongside massively parallel computing techniques to address efficiency bottlenecks. The seamless incorporation of analytical tools with BI systems will play a key role in managing the complexity of Big Data (Adewusi et al. 2024).

The use of data warehouses and data lakes within BI systems allows organizations to analyze extensive volumes of structured and unstructured data, thereby supporting more informed decision-making. NLP’s potential as a non-human innovation intermediary also presents new opportunities to synthesize

vast information landscapes, which organizations can leverage to enhance efficiency and stay ahead of emerging trends and consumer needs (Nambiar & Mundra 2022; Just 2024). These capabilities further emphasize the strategic importance of integrating NLP solutions into BI workflows.

Continued advancements in transformer-based model training and refined methodologies will be essential to improve the performance of NLP models in diverse BI applications (Kotei & Thirunavukarasu 2023).

Categorizing BI applications and user interactions offers a structured approach to identify areas for improvement, ensuring NLP-driven BI systems meet a range of user needs and application scenarios (Passlick et al. 2023). Further research should explore ethical considerations, such as bias mitigation, to ensure NLP systems serve users equitably. Additionally, understanding data governance within data lakes and warehouses can enhance data quality and reliability in NLP systems (Nambiar & Mundra 2022). These aspects highlight the critical need for ongoing research to address ethical considerations and biases in NLP models as they become integral to decision-making processes. Future studies should aim to develop frameworks for transparency and equity in deploying NLP in BI systems, ensuring these technologies responsibly serve diverse user needs (Just 2024).

In summary, the findings from the literature highlight promising directions for advancing NLP in BI while emphasizing the importance of addressing key challenges. By focusing on scalability, ethical considerations, methodological advancements, and robust data integration practices, future research can pave the way for more effective and equitable NLP-driven BI systems.

3 Research Methods and Tools

This chapter outlines the systematic approaches, tools, and techniques used to prepare for the integration of Natural Language Processing (NLP) with Business Intelligence (BI) systems. The processes include dataset preparation, model selection, and fine-tuning to support automated query workflows.

3.1 Tools and Technologies

The implementation of this study required an integrated set of tools and technologies to ensure efficient data handling, fine-tuning, system development, query execution, and business intelligence reporting.

3.1.1 Business Intelligence Platform

Power BI was employed for presenting query results through dynamic dashboards, offering robust tools for filtering, analysis, and automated data refresh. Power Query, integrated within Power BI, facilitated data transformation tasks such as cleaning and formatting outputs retrieved from the PostgreSQL database before visualization

3.1.2 Development Environment

Jupyter Notebook served as the primary environment for exploratory data analysis (EDA), dataset preprocessing, SQL query prototyping, and model development. Its interactive nature enabled iterative experimentation, real-time visualization of results, and seamless transitions between project stages.

3.1.3 Programming Language and Libraries

Python was the main programming language, chosen for its versatility and extensive ecosystem. Pandas and NumPy supported structured data manipulation, numerical computations, and statistical analysis. The OS library managed environment variables and file handling, while Hashlib, Random, and String ensured secure data anonymization using hashing techniques and salt generation. The Datetime library facilitated the extraction and analysis of temporal components, essential for time-based queries and trends.

For handling large-scale data, Dask was utilized to process datasets exceeding memory limitations. Its parallelized operations allowed efficient computations and data transformations, while Dask-ML provided scalable train-test splits. Scikit-learn further supported dataset management and evaluation, contributing tools for splitting data into training and test sets and calculating metrics like accuracy, precision, recall, and F1-score.

3.1.4 Machine Learning and NLP Frameworks

The deep learning framework PyTorch was used to fine-tune the chosen NLP models for the natural language-to-SQL query task. PyTorch provided GPU acceleration capabilities using CUDA, enabling faster model training and inference when GPU resources were available. Custom training loops implemented gradient clipping for numerical stability and early stopping mechanisms to halt training once validation performance plateaued.

The Hugging Face Transformers library was critical for accessing pre-trained NLP models like GPT-2, T5 (Text-to-Text Transfer Transformer), and BERT. These models were fine-tuned to generate SQL queries from natural language inputs. SQL-specific keywords, such as SELECT, FROM, and WHERE, were integrated into the tokenizer to ensure compatibility with SQL syntax. Techniques like beam search decoding were employed to improve the quality of generated SQL queries, ensuring accuracy and relevance.

Custom PyTorch Dataset classes and DataLoaders were implemented to efficiently preprocess and batch natural language and SQL query pairs for model training. The Hugging Face Datasets library further streamlined dataset loading and management, facilitating smooth integration with the Transformers pipeline.

3.1.5 SQL Query Tools and Database Management

PostgreSQL was the core relational database used to store preprocessed datasets and query results. Its advanced SQL functionalities ensured reliable data storage, retrieval, and query execution. SQLAlchemy, acting as an Object Relational Mapper (ORM), enabled secure and parameterized interactions with PostgreSQL, preventing SQL injection risks. Psycopg2 provided efficient Python-to-PostgreSQL connectivity for data transfer and query execution.

pgAdmin served as the interface for database administration, allowing for the validation, execution, and debugging of SQL queries during development. To maintain syntactic correctness and improve query readability, SQLParse and SQLGlot were used for formatting, parsing, and validating SQL statements.

3.1.6 Visualization Tools

For exploratory data analysis and visualizing patterns, Matplotlib and Seaborn were employed. Matplotlib generated foundational static plots such as histograms, scatter plots, and distributions, while Seaborn provided advanced statistical visualizations, including heatmaps, KDE plots, and bar charts. These tools enabled in-depth analysis of data distributions, relationships, and trends before feeding the data into machine learning models.

3.1.7 Privacy and Security Tools

To safeguard sensitive fields like recipient and manufacturer IDs, secure hashing was implemented using Hashlib, Random, and String. Environment variables for database credentials were managed securely using Python's OS library.

3.1.8 System Development Frameworks

For backend development of the final solution, FastAPI was employed to implement a lightweight and asynchronous API. It processed user inputs, invoked fine-tuned models, and executed SQL queries against the PostgreSQL database, ensuring efficient real-time operations. On the frontend, Streamlit provided an intuitive web-based interface for user interaction. Users could input natural language queries, select models, and view the resulting SQL queries and outputs seamlessly. The simplicity of Streamlit enabled quick deployment of a user-friendly frontend without the need for extensive development.

3.2 Dataset Selection and Preparation

The foundation of this research is a dataset that supports the exploration of NLP-driven automation in Business Intelligence (BI) systems. This dataset serves as the basis for generating and testing SQL queries, creating BI dashboards, and conducting analyses in Power BI. Selecting a dataset that reflects the typical characteristics of business intelligence data, such as structured, voluminous, and rich in attributes, was essential to ensuring its relevance to the goals of this study.

With these requirements in mind, the CMS Open Payments, General Payment Data 2023 dataset was chosen. This dataset provides a robust framework for analyzing the integration of NLP into BI workflows and demonstrates how query automation can make BI systems more accessible and efficient.

3.2.1 Dataset Justification and Relevance

The CMS Open Payments, General Payment Data 2023 dataset was selected for its comprehensive structure, significant size, and relevance to the healthcare domain. Publicly available through the Centers for Medicare and Medicaid Services (CMS), it comprises over 14.6 million records detailing financial transactions, such as consulting fees, research funding, and travel reimbursements. This rich dataset provides an ideal foundation for exploring financial trends and relationships, which is essential for business intelligence (BI) research.

The dataset, published under the U.S. federally mandated Open Payments program, promotes transparency in healthcare by publicly disclosing financial relationships. This aligns with BI's goal of enhancing decision-making frameworks, as emphasized in the literature, where structured and voluminous datasets are crucial for deriving actionable insights in dynamic fields like healthcare (Tavera Romero et al., 2021). The static nature of the dataset, covering the year 2023, enables a focused analysis, bypassing the complexities of real-time data synchronization. This simplicity supports the integration of NLP into BI systems, allowing natural language queries to democratize data access and eliminate technical barriers, as highlighted by Al-Okaily et al. (2023).

Beyond its practicality, the dataset supports diverse BI tasks, including trend analysis and payment distribution studies, making it well-suited for evaluating NLP-driven tools. The literature underlines the value of datasets like this, which combine complexity and structure, in advancing BI accessibility and effectiveness (Paaß & Giesselbach, 2023). Challenges, such as missing values and outliers, further mirror real-world BI obstacles, providing a robust testbed for user-centric workflows that integrate pre-trained NLP models (Kotei & Thirunavukarasu, 2023).

In summary, the CMS Open Payments dataset serves as both a rich resource for healthcare financial analytics and an exemplary platform for advancing NLP applications in BI systems. It bridges theoretical BI concepts with practical

challenges, supporting the development of tools that enhance data-driven decision-making.

3.2.2 Exploratory Data Analysis (EDA)

To assess the suitability of the CMS Open Payments dataset for this research, a detailed exploratory data analysis (EDA) was conducted using Jupyter Notebook with tools such as Dask and Pandas. This analysis examined the dataset’s structure, identified challenges, and uncovered patterns relevant to BI system development.

The dataset includes 14,609,233 rows and 91 columns detailing financial transactions, with mixed content handled through default data types and explicit casting of key numerical columns like “Total Amount of Payment (USD)” to ensure computational accuracy. Substantial missing values in fields such as “Teaching Hospital Name” and “Covered Recipient NPI” were observed, reflecting real-world imperfections that BI systems must address. The dataset also showcased diverse categorical attributes, with over 15,000 unique entries in "Recipient City" and 300 categories in "Covered Recipient Specialty," supporting a wide range of analytical use cases.

Outlier detection using Z-score and interquartile range (IQR) methods revealed anomalies in payment amounts and counts, emphasizing the need for robust processing to ensure accuracy in BI dashboards and NLP query results. Numerical columns displayed significant right skewness, particularly in payment amounts, dominated by small transactions with occasional large sums. Addressing this skewness was critical for normalizing data distributions during query automation.

Correlation analysis showed minimal linear relationships between numerical attributes, underscoring the dataset’s complexity and the opportunity for NLP models to uncover non-linear patterns. These findings highlighted the dataset's alignment with real-world BI challenges, such as missing data, outliers, and

skewed distributions, providing a robust foundation for testing NLP-driven query automation.

In summary, the EDA validated the CMS Open Payments dataset as a rich and structured environment for exploring BI query automation.

3.2.3 Dataset Preprocessing

The preprocessing phase was crucial in transforming the CMS Open Payments, General Payment Data 2023 dataset into a clean, structured, and anonymized form suitable for integration into a Business Intelligence (BI) framework. This stage ensured the dataset's quality and relevance for exploring query automation and enhancing BI system functionality.

The dataset, comprising over 14.6 million rows, was processed using Dask for its ability to handle large-scale data efficiently. From the initial 91 columns, 30 were retained based on their relevance to financial, contextual, and relational insights. Attributes such as names and other personal identifiers were excluded to prioritize privacy and maintain an aggregated level of analysis. To improve usability, column names were simplified, such as renaming Total_Amount_of_Payment_USDollars to Payment Amount (USD), balancing clarity and user accessibility.

Addressing missing values was a key focus. Columns with over 50% missing data, such as Charity and Device PDI, were removed to enhance dataset consistency and reduce computational overhead. Critical fields like Recipient Profile ID and Recipient NPI, essential for accurate anonymization, had incomplete rows removed. Missing categorical values, such as those in City and State, were replaced with "Unknown," preserving dataset structure. Numeric fields, including Payment Amount (USD), were filled with zeros as neutral placeholders to prevent analytical disruptions. Remaining gaps in categorical fields were filled with the mode, ensuring consistency without introducing artificial variability.

Sensitive identification fields were anonymized to uphold ethical data handling standards. Recipient Profile ID, Recipient NPI, Manufacturer Payment ID, and Manufacturer Payment Name were replaced with hashed equivalents, combining random salts with the original values. This anonymization preserved the dataset's analytical utility while safeguarding sensitive information and complying with privacy guidelines.

Feature engineering further enriched the dataset. Temporal attributes, such as year, month, and day, were extracted from Payment Date and Publication Date fields to support trend analyses. A new Contextual Description column was created by combining key fields, providing concise summaries of payment contexts. Highly skewed numeric fields, such as Payment Amount (USD) and Payment Count, underwent log transformations, normalizing their distributions for improved compatibility with statistical methods and machine learning models.

Duplicate entries in key identifiers revealed the need for unique transaction tracking. A new Transaction ID column was added, assigning unique values to each row and ensuring data integrity for BI workflows. The dataset index was reset to stabilize partitions and maintain consistency during these transformations.

Following preprocessing, the dataset was split into training and testing subsets, with 20% allocated for testing. This division ensured reliable evaluation of machine learning workflows. The preprocessed training and testing subsets were first loaded into separate tables within the PostgreSQL database. Subsequently, the full preprocessed dataset was loaded as another table. This integration supports the study's goal of enhancing BI accessibility through automated query workflows while ensuring the dataset remains consistent, organized, and accessible for analysis.

This comprehensive preprocessing pipeline ensured that the dataset was meticulously prepared for exploring NLP-driven query automation in BI systems. By addressing real-world data challenges, anonymizing sensitive information,

and enriching the dataset with engineered features, it now serves as a robust foundation for advanced analysis. These efforts not only enhance the dataset's analytical value but also align it with the study's overarching goal of making BI systems more accessible, efficient, and user-friendly.

3.3 Database Integration

The integration of the preprocessed CMS Open Payments dataset into a PostgreSQL database marked a pivotal step in supporting the study's objective of automating BI query workflows.

3.3.1 Database Creation

To facilitate the integration of the preprocessed CMS Open Payments dataset, a PostgreSQL database was manually created using the pgAdmin interface. This database serves as the centralized repository for the dataset in its various forms. The full preprocessed dataset was stored in a table named Preprocessed_Dataset, while the training and testing subsets were stored in two separate tables: Preprocessed_Train_Data and Preprocessed_Test_Data. The table structures were dynamically generated during the data loading process via Python's SQLAlchemy library and the Pandas to_sql method, ensuring seamless mapping of the dataset's schema into PostgreSQL while maintaining consistency with the preprocessed data structure. This organization ensures that the full dataset is readily accessible alongside the training and testing subsets, supporting both exploratory analysis and machine learning workflows while aligning with the study's goal of enhancing BI accessibility.

PostgreSQL's robust scalability and compatibility with complex querying requirements make it an ideal choice for managing datasets of this magnitude. As noted in the literature, databases that support advanced querying and structured data management are critical for enabling automated workflows in BI systems.

3.3.2 Validation of Data Integration

To ensure the integrity and accuracy of the foundational dataset following its integration into PostgreSQL, a comprehensive validation process was conducted using SQL queries executed in the pgAdmin Query Tool. This step confirmed the successful transfer of data from preprocessing to the database and verified that the tables aligned with the intended structure and data quality requirements. The same validation processes were, of course, also applied to the Preprocessed_Dataset table to ensure consistency across all database tables.

Validation began with verifying the total row counts in the Preprocessed_Train_Data, Preprocessed_Test_Data, and Preprocessed_Dataset tables. The row counts were compared with the preprocessed dataset to ensure no records were lost during the data loading process. This step guaranteed the completeness of the database, providing a reliable basis for analysis.

The uniqueness of the Transaction ID column, introduced during preprocessing, was also validated in all three tables. Ensuring that each record had a unique identifier was essential for maintaining data integrity and supporting BI workflows reliant on precise transaction tracking.

Further validation involved analyzing the distribution of key categorical fields, such as Recipient Type and Manufacturer Name, in each table. These distributions were compared with expectations from preprocessing to confirm that no significant deviations occurred during data integration. This step safeguarded the dataset's consistency across critical analytical dimensions.

Handling of missing values was another focus area. Placeholder values, such as "Unknown" for categorical fields and 0 for numeric fields, were validated to confirm their consistent application throughout all three tables. These placeholders ensured the datasets were complete and ready for analysis without gaps that could disrupt workflows.

Finally, the data types of critical columns in the Preprocessed_Train_Data, Preprocessed_Test_Data, and Preprocessed_Dataset tables were validated against the expected schema. This step ensured that numeric, categorical, and temporal fields adhered to their correct formats, maintaining compatibility with downstream analytical tools. Cross-checking data types also provided confidence in the datasets' readiness for automated querying and advanced BI processes.

3.4 Power BI Integration

The integration of the foundational dataset into Power BI enabled the visualization and analysis of data to support enhanced business intelligence (BI) insights. The Preprocessed_Train_Data, Preprocessed_Test_Data, and Preprocessed_Dataset tables were directly connected to PostgreSQL, establishing a seamless link between the database and the visualization platform. This setup ensured that all variations of the preprocessed data were readily available for creating dynamic and interactive visualizations to support comprehensive BI workflows.

3.4.1 Connecting PostgreSQL to Power BI

The connection between PostgreSQL and Power BI was established by configuring PostgreSQL as the data source. This process involved navigating to Power BI's "Get Data" feature, selecting PostgreSQL as the source, and entering the database credentials. Once authenticated, the Preprocessed_Train_Data, Preprocessed_Test_Data, and Preprocessed_Dataset tables were loaded into Power BI's workspace, enabling direct access for exploration and visualization. All three tables were integrated within the same workspace, allowing for consistent analysis across the datasets.

3.4.2 Data Validation and Exploration

Once the connection between PostgreSQL and Power BI was established, the dataset was explored in Power BI's "Report View," where selecting columns from the field pane automatically generated visualizations such as bar charts, histograms, and summaries. This functionality offered an efficient and visually engaging way to examine data patterns, relationships, and potential irregularities. Tailored visualizations were also developed as needed to delve deeper into specific attributes and trends.

Viewing the data in this clear, interactive format brought previously unnoticed discrepancies to the forefront. For example, the City column contained invalid entries with numerical or special characters, and the Zip Code column exhibited inconsistent formatting, with some entries restricted to the first five digits while others included an extended format. Additionally, the Publication Date column was found to contain a single static value (2024-06-28), reducing its relevance for analytical purposes.

To address these discrepancies, SQL scripts were executed in pgAdmin to clean and standardize the data. Invalid entries in the City column were removed, and unnecessary spaces were trimmed. The Zip Code column was standardized to ensure uniformity by retaining only the first five digits. Furthermore, the Publication Date column and its derived fields—Publication Year, Publication Month, and Publication Day—were removed due to their limited analytical value. These corrections were implemented consistently across the Preprocessed_Train_Data, Preprocessed_Test_Data, and Preprocessed_Dataset tables to maintain alignment.

Following these adjustments, the refined tables were reconnected to Power BI, and the data was refreshed. Subsequent exploration verified that the updates were accurately reflected in the visualizations. During this process, it was also noted that some fields, including Program Year and Transaction ID, were treated as numeric data and automatically aggregated. To address this, these

fields were converted to text data types, and all summary fields were adjusted to “Do Not Summarize,” ensuring appropriate handling of the dataset.

This iterative process of visual validation and refinement enhanced the dataset’s quality and consistency. The ability to identify and resolve these discrepancies in an interactive and visual context significantly strengthened the dataset’s reliability.

3.4.3 Dashboard Creation

Building upon the integration of the dataset into Power BI and the data validation process, a dashboard (Figure 1) was created to explore the dataset further and prototype visualizations for potential use in the final solution. Using the training portion of the dataset ensured analytical depth while maintaining computational efficiency. This approach enabled iterative development without overwhelming the system, while still reflecting the dataset's key characteristics.

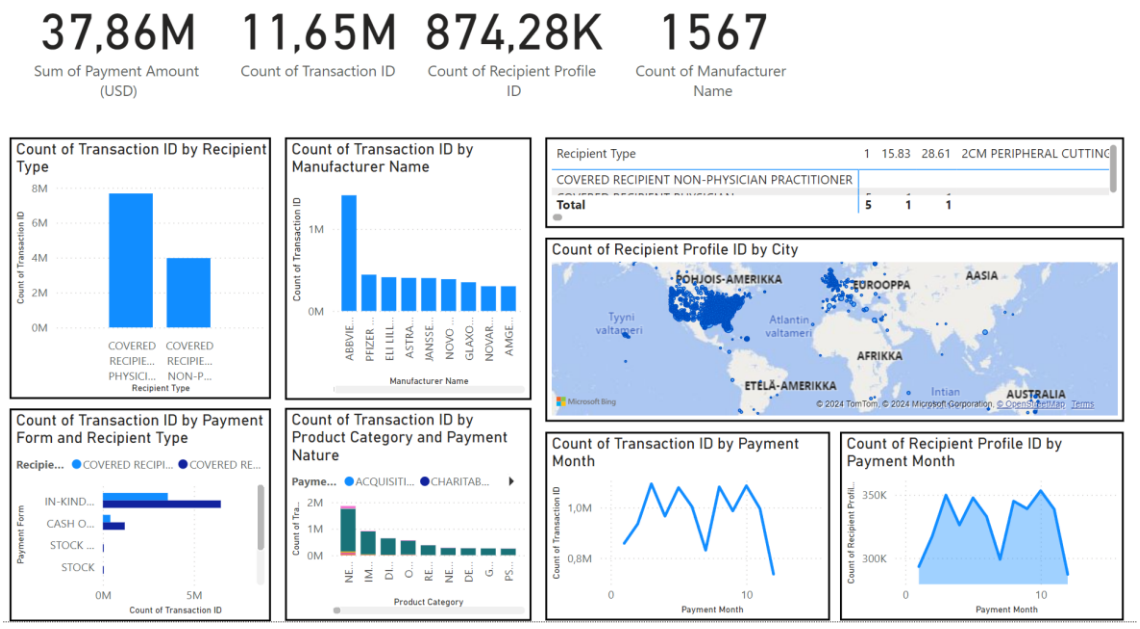

Figure 1. Power BI dashboard for data exploration and prototyping.

The dashboard was designed to deliver actionable insights across key dimensions, emphasizing relationships between recipients, manufacturers, products, payment forms, and temporal trends. Each visualization was chosen to highlight critical aspects of the dataset. While the dashboard as a whole was not intended for direct implementation in the final solution, it provided a stepping stone to identify specific visualizations for integration into the final framework.

The recipient and manufacturer analysis section clarified interactions between these central entities. A clustered bar chart displayed transaction counts by recipient type, showcasing dominant categories such as physicians or teaching hospitals and revealing trends across groups. A tree map visualized transaction counts by manufacturer, offering a hierarchical view that highlighted major contributors while allowing exploration of smaller entities. Together, these visualizations provided a balanced perspective on the dataset's core participants.

For product and payment form trends, visualizations captured transaction dynamics across categories. A clustered bar chart analyzed payment forms by recipient type, revealing associations between specific groups and payment types. A stacked column chart highlighted relationships between product categories and payment natures—such as consulting fees or research funding—offering layered insights. Additionally, a matrix visualization mapped recipient types against product categories, enabling users to detect patterns and cross-dimensional relationships.

Geographic and temporal trends were addressed through spatial and time-based analyses. A map visualization illustrated distinct recipients across states, shedding light on regional disparities in activity. For temporal analysis, a line chart tracked transaction counts by month, revealing seasonal patterns or fluctuations. Together, these visualizations provided a comprehensive view of the dataset’s regional and temporal characteristics.

Guided by the principle of maximizing insight through simplicity, the dashboard featured visualizations that revealed significant patterns and relationships while remaining accessible and easy to interpret.

3.5 Natural Language and Query Pair Generation

For both fine-tuning and post-fine-tuning evaluation of the models, it was essential to design query pairs consisting of a natural language question or prompt and its corresponding SQL query. These query pairs served as the backbone of the fine-tuning and evaluation processes, ensuring that the models could effectively transform user-friendly natural language inputs into actionable SQL queries.

3.5.1 Designing Query Pairs

The process of generating natural language and SQL query pairs began with a thorough review of the dataset using Power BI. This review was vital for understanding the dataset's structure, including its column names and sample values, and served as a foundation for crafting practical and realistic queries. The primary objective was to ensure that the SQL equivalents of these queries could be executed directly in PostgreSQL, reflecting real-world applications without requiring modifications.

For fine-tuning the model, a total of 250 query pairs were meticulously created (Table 1). Natural language queries were phrased without explicitly mentioning table names, while their corresponding SQL queries consistently referenced the Preprocessed_Dataset table, representing the entire dataset. This approach ensured that the model was trained to query the full dataset, reflecting its intended use case.

To expand the initial set of query pairs, ChatGPT, a generative AI language model developed by OpenAI, was utilized. As a transformer-based model, ChatGPT is designed for a wide range of natural language understanding and

generation tasks, making it an effective tool for generating structured query pairs. Contextual information about the dataset and examples of existing query pairs were provided, enabling ChatGPT to generate additional pairs in the required format. These generated pairs underwent a rigorous review and refinement process to ensure their accuracy, consistency, and adherence to PostgreSQL standards.

Table 1. Examples from the set of 250 fine-tuning query pairs.

Natural Language Query	**Corresponding SQL Query**
What is the total payment amount in 2023?	SELECT SUM("Payment Amount (USD)") FROM "Preprocessed_Dataset" WHERE "Program Year" = 2023;
Calculate the average payment amount per transaction.	SELECT AVG("Payment Amount (USD)") FROM "Preprocessed_Dataset";
How many transactions are recorded in the dataset?	SELECT COUNT("Transaction ID") FROM "Preprocessed_Dataset";
How many distinct recipients are there in the dataset?	SELECT COUNT(DISTINCT "Recipient Profile ID") FROM "Preprocessed_Dataset";

In addition to the fine-tuning queries, a separate set of 25 distinct query pairs was created for evaluation (Table 2). To maintain consistency with the fine-tuning set, these evaluation queries were also structured to reference the Preprocessed_Dataset table in their SQL equivalents. ChatGPT was again utilized to generate these evaluation pairs, following explicit instructions to ensure they were entirely distinct from the fine-tuning set. A Python script, executed within Jupyter Notebook, validated the evaluation queries to confirm there was no overlap in phrasing or structure with the fine-tuning pairs. Once verified, the evaluation queries were organized in the same format as the fine-tuning set, facilitating a structured evaluation of the model’s ability to generalize to new, unseen queries.

Table 2. Examples from the set of 25 evaluation query pairs.

Natural Language Query	**Corresponding SQL Query**
What is the total payment amount across all states for the year 2023?	SELECT SUM("Payment Amount (USD)") FROM "Preprocessed_Dataset" WHERE "Program Year" = 2023;
What is the average payment amount per transaction in California in 2023?	SELECT AVG("Payment Amount (USD)") FROM "Preprocessed_Dataset" WHERE "State" = 'CA' AND "Program Year" = 2023;
What is the total payment amount for food and beverages in Texas in 2023?	SELECT SUM("Payment Amount (USD)") FROM "Preprocessed_Dataset" WHERE "State" = 'TX' AND "Payment Nature" = 'FOOD AND BEVERAGE';
How many unique recipient profiles are there in New York for 2023?	SELECT COUNT(DISTINCT "Recipient Profile ID") FROM "Preprocessed_Dataset" WHERE "State" = 'NY' AND "Program Year" = 2023;

3.5.2 Validating Query Pairs

To ensure the SQL queries were usable for querying in PostgreSQL, a validation process was conducted using a Python script in Jupyter Notebook and the pgAdmin query tool. The validation began with a script that iterated through the provided query pairs. For each pair, the SQL queries were cleaned by stripping unnecessary escape characters, such as converting \'CA\' to 'CA'. This step ensured the queries adhered to PostgreSQL syntax requirements.

Once cleaned, the script extracted all the SQL queries from the query pairs and formatted them into a copyable structure. These extracted queries were then manually tested in the pgAdmin query tool. Each query was executed to confirm it ran successfully and returned the expected results. Any errors or issues were corrected, ensuring all queries were functional.

The validation process was conducted for both the fine-tuning queries and the evaluation queries, guaranteeing that all SQL statements were accurate, executable, and suitable for use in the PostgreSQL environment.

3.6 NLP Model Selection

The selection of T5, BERT, and GPT for this study was guided by their individual strengths and suitability for Natural Language Processing (NLP) tasks, particularly in the areas of query transformation and comprehension. As highlighted in the literature review, these models are recognized as leading approaches in NLP, each offering distinct advantages.

BERT was chosen for its ability to understand contextual relationships within text, making it suitable for classifying and interpreting natural language queries (Paradza & Daramola 2021). T5's sequence-to-sequence architecture offered significant advantages for generating SQL queries from natural language inputs, aligning closely with the study's objectives (Wang et al. 2023). GPT was considered for its generative capabilities but ultimately proved less effective due to its limitations in producing structured outputs like SQL (Just 2024).

3.7 Fine-Tuning and Evaluating NLP Models

To study how NLP can be applied to automate SQL generation for business intelligence, it was necessary to fine-tune the chosen NLP models: T5, BERT, and GPT, for this specific task. The fine-tuning and evaluation processes were deliberately standardized to ensure fairness and comparability. Python served as the programming language, while Jupyter Notebook was the implementation environment, offering flexibility and transparency throughout the process. The core fine-tuning structure, including training loops, evaluation scripts, and early stopping criteria, remained largely the same for all models, with minor adjustments to account for their architectural differences.

To maintain consistency, each model was fine-tuned at three learning rates (1e-5, 3e-5, and 5e-5), ensuring equal effort and resources were dedicated to each. This systematic approach avoided the need for exhaustive hyperparameter tuning for individual models, which would have been time-intensive and counter to the thesis's goal of providing a balanced evaluation. Instead, the emphasis

was on applying the same level of refinement across all models to facilitate an objective comparison of their capabilities.

Despite the shared methodology, the models required specific adaptations due to their different architectures. T5, a sequence-to-sequence model designed for text generation tasks, was fine-tuned to generate SQL queries directly from natural language inputs. BERT, a bidirectional encoder optimized for classification tasks, was adapted to predict cluster labels corresponding to SQL query groups. GPT, a generative autoregressive model, was fine-tuned to complete SQL queries based on natural language prompts.

3.7.1 Fine-Tuning T5

The fine-tuning process of the T5 model, which served as a blueprint for the fine-tuning of the other models, was conducted across three separate notebooks, each systematically designed to build and evaluate fine-tuned versions of the model using varying learning rates. These notebooks focused on adapting the pre-trained T5 model to the specific task of translating natural language queries into SQL. The objective was to produce a model capable of handling the complexities and nuances of SQL query generation while addressing domain-specific requirements.

The dataset used for fine-tuning consisted of a comprehensive collection of 250 natural language and SQL query pairs. To ensure robust evaluation, this dataset was split into training and testing sets, allocating 80% (200 query pairs) for training and 20% (50 query pairs) for evaluation. Each notebook implemented a distinct learning rate (1e-5, 3e-5, and 5e-5 respectively), allowing for an exploration of how this hyperparameter influenced the model's training dynamics and overall performance.

In the first notebook, a learning rate of 1e-5 was used for fine-tuning. This relatively low learning rate enabled the model to make gradual adjustments during training, reducing the likelihood of overshooting optimal parameter values. The second notebook increased the learning rate to 3e-5, aiming for a

balance between training stability and convergence speed. Finally, the third notebook utilized a higher learning rate of 5e-5 to accelerate the convergence process, albeit with a potentially increased risk of instability or overfitting.

Each notebook followed a structured workflow. Initially, the Hugging Face transformers library was used to load the T5 tokenizer and model, setting the foundation for handling this sequence-to-sequence task. The dataset was then processed into the T5 format, where the natural language query served as the input and the corresponding SQL query acted as the target. A custom PyTorch Dataset class was implemented to tokenize these inputs and targets, ensuring compatibility with the T5 architecture. This implementation also handled padding and truncation to a fixed maximum sequence length. For efficient training and evaluation, the data was batched using PyTorch's DataLoader, which shuffled training batches to improve generalization and prepared test batches for evaluation.

The fine-tuning loop was meticulously designed to optimize model performance while maintaining training stability. The AdamW optimizer was employed, which is well-suited for transformer models due to its treatment of weight decay. Gradient clipping was applied to prevent exploding gradients, ensuring stable updates to the model parameters. The training loop computed the cross-entropy loss at each step, using teacher forcing to encourage the model to closely mimic the target SQL query. Early stopping was implemented, halting training if validation loss did not improve over three consecutive epochs. This prevented overfitting and ensured efficient utilization of computational resources.

To monitor the model's progress, training and validation losses were recorded after every epoch. Whenever the validation loss improved, the model's state was saved, ensuring that the best-performing version was preserved for further use. These saved models captured the weights that minimized validation loss, providing checkpoints for subsequent evaluation.

The results of the fine-tuning process demonstrated consistent reductions in validation loss across all three learning rates, indicating successful adaptation

of the model to the query translation task. The notebook utilizing a learning rate of 1e-5 achieved a final validation loss of 0.1082, reflecting the stability and precision of this lower learning rate. The model trained at 3e-5 reached a slightly higher final validation loss of 0.1229, which nevertheless represented strong performance. The model fine-tuned at 5e-5 exhibited faster convergence, with validation loss dropping to 0.1263 by the 42nd epoch. However, the higher variability inherent to this learning rate was reflected in marginally higher loss compared to the other two models.

The fine-tuned models were evaluated on the reserved test set of 50 query pairs, where metrics such as accuracy, precision, recall, and F1 score were calculated. The model fine-tuned with a learning rate of 1e-5 achieved an accuracy of 0.4400, with precision, recall, and F1 scores matching this value. The model trained with a learning rate of 3e-5 achieved an accuracy of 0.4000, along with equal precision, recall, and F1 scores. The model fine-tuned with a learning rate of 5e-5 demonstrated an accuracy of 0.3600, with the same values for precision, recall, and F1. These results (Table 3) highlighted that the models handled simpler queries effectively, often producing outputs that were both syntactically and semantically correct. For example, queries such as “How many transactions are recorded in the dataset?” were consistently translated into the correct SQL syntax.

Table 3. Evaluation metrics for fine-tuned T5.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
T5 finetuned with learning rate 1e-5	0.40	0.40	0.40	0.40
T5 finetuned with learning rate 3e-5	0.34	0.34	0.34	0.34
T5 finetuned with learning rate 5e-5	0.36	0.36	0.36	0.36

While these metrics provide a quantitative assessment of the models’ performance, they do not tell the full story. SQL is inherently flexible, allowing

the same logical operations to be expressed in multiple syntactically different ways. For instance, the models occasionally generated queries that, while not exact matches to the reference, were still functionally equivalent. This limitation of the evaluation metrics underscores the importance of qualitative inspection alongside quantitative measures.

Despite these challenges, all three models demonstrated substantial capability in translating natural language to SQL, with the lower learning rates (1e-5 and 3e-5) showing better overall performance in terms of loss and evaluation metrics. The model trained with 5e-5, although slightly less accurate in handling complex queries, achieved rapid learning during initial epochs, making it a valuable addition to the comparative analysis.

Finally, the fine-tuned T5 models were saved to designated directories, each labeled with its corresponding learning rate.

3.7.2 Fine-Tuning BERT

The fine-tuning process for BERT built upon the structured approach established in the T5 fine-tuning workflow, with necessary adaptations made to suit BERT's architecture. The T5 Jupyter Notebook served as a foundational blueprint, ensuring consistency across the fine-tuning methods while accommodating the specific requirements of each model. Like T5, BERT was fine-tuned on the task of translating natural language queries into SQL; however, this required reimagining the task as a classification problem due to BERT's bidirectional encoder architecture.

To adapt BERT for the task, the dataset of 250 natural language and SQL query pairs, split into 200 training and 50 testing pairs, was preprocessed to represent SQL queries as cluster labels. This approach enabled the model to predict which cluster, or group, a given natural language query belonged to, simplifying the SQL generation process into a classification task. The cluster labels were derived using k-means clustering, where each SQL query was represented as a vector in latent space using sentence embeddings from a pre-trained language

model. This step was implemented using the sentence-transformers library. Each natural language query was assigned to the same cluster as its corresponding SQL query, ensuring consistent alignment between input and output data.

As with T5, the fine-tuning process involved training three versions of the model using learning rates of 1e-5, 3e-5, and 5e-5 to explore the effect of this hyperparameter on model performance. The Hugging Face transformers library was used to load the pre-trained BERT base model and tokenizer. The tokenizer encoded natural language queries into input embeddings suitable for the BERT architecture. Since BERT is designed for token-level tasks, a classification head was added to the pre-trained encoder to predict cluster labels.

The fine-tuning loop, like in T5, utilized the AdamW optimizer for efficient weight updates, with gradient clipping applied to stabilize training. Cross-entropy loss was computed to measure the discrepancy between the predicted cluster probabilities and the true cluster labels. Early stopping was implemented to prevent overfitting, with training halted if validation loss did not improve for three consecutive epochs.

Evaluation metrics included accuracy, precision, recall, and F1 score, calculated using the reserved test set of 50 natural language-SQL pairs. For BERT, these metrics assessed the model's ability to classify natural language queries into the correct SQL clusters. The cluster-based evaluation introduced some limitations, as it relied on clustering quality, which in turn depended on the k-means algorithm's performance and the choice of k. However, this approach provided a scalable and interpretable method for adapting BERT to SQL generation.

The fine-tuning results revealed that BERT achieved competitive performance across the three learning rates. The model trained at 1e-5 achieved the highest accuracy of 0.4200, with precision, recall, and F1 scores matching this value. At a learning rate of 3e-5, the model achieved slightly lower accuracy of 0.4000,

with similar values for precision, recall, and F1. The model fine-tuned with 5e-5 exhibited the fastest convergence but showed reduced stability, resulting in an accuracy of 0.3800.

Table 4. Evaluation metrics for fine-tuned BERT.

Model	Accuracy	Precision	Recall	F1 Score
BERT finetuned with learning rate 1e-5	0.86	0.8817	0.86	0.8477
BERT finetuned with learning rate 3e-5	0.86	0.8514	0.86	0.8406
BERT finetuned with learning rate 5e-5	0.84	0.8581	0.84	0.8318

Despite the classification approach simplifying SQL generation, BERT successfully captured meaningful relationships between natural language queries and SQL clusters. However, the reliance on clustering meant that BERT occasionally misclassified queries that belonged to overlapping or ambiguous clusters. For example, natural language queries with similar intent but slightly different wording were sometimes assigned to different clusters, affecting precision and recall.

The fine-tuned BERT models were saved to designated directories corresponding to their learning rates, ensuring traceability and enabling further experimentation. Overall, the fine-tuning process demonstrated BERT's adaptability to SQL query generation, with the clustering-based approach offering a viable alternative to direct sequence-to-sequence modeling for structured tasks.

3.7.3 Fine-Tuning GPT

The fine-tuning process for GPT followed the standardized methodology applied to T5 and BERT to ensure fairness and comparability. As with the other models, the goal was to adapt GPT, a generative autoregressive model, to translate

natural language queries into SQL statements. While the underlying structure of the fine-tuning pipeline remained the same, specific adjustments were made to accommodate GPT's architecture.

The dataset, similarly to T5 and BERT, consisted of 250 natural language and SQL query pairs, and was split into 200 training examples and 50 test examples to facilitate robust evaluation. This ensured that the training and evaluation phases remained consistent across all models, enabling a direct and fair comparison of their performance. Fine-tuning was conducted across three separate notebooks, with learning rates of 1e-5, 3e-5, and 5e-5, reflecting the same approach applied to T5 and BERT. This systematic exploration of learning rates ensured that equal effort and resources were devoted to GPT as to the other models.

For GPT fine-tuning, the Hugging Face transformers library was used to load the pre-trained GPT-2 model and tokenizer. The tokenizer was extended with special tokens, including padding tokens and SQL-specific keywords such as SELECT, FROM, WHERE, and GROUP BY, to better accommodate the syntax and structure of SQL queries. The embedding layer of the GPT-2 model was resized to account for these newly added tokens. Given GPT's autoregressive nature, padding was applied to the left side of the input sequences to ensure compatibility during training. The task was framed as text generation, where the model was trained to predict the entire SQL query given a natural language prompt.

A custom PyTorch Dataset class was implemented to preprocess the inputs and outputs. Natural language queries were tokenized as inputs, and corresponding SQL queries were tokenized as targets. Padding and truncation ensured consistent sequence lengths, maintaining compatibility with GPT's architecture. During training, GPT was optimized using its causal language modeling objective, where it was tasked with predicting the next token in the sequence. The data was batched using PyTorch's DataLoader with a batch size of 8, balancing computational efficiency and model performance.

The fine-tuning process itself adhered closely to the workflow established for T5 and BERT. The AdamW optimizer was employed with learning rates of 1e-5, 3e-5, and 5e-5. Gradient clipping was applied to stabilize training, and early stopping was used to halt the process if the validation loss did not improve for three consecutive epochs. This approach ensured that computational resources were utilized efficiently and mitigated the risk of overfitting. Training and validation losses were monitored after every epoch, and the best-performing model state was saved whenever validation loss improved.

Despite this structured and systematic fine-tuning process, GPT's performance was profoundly disappointing. The evaluation phase, conducted on the reserved test set of 50 examples, revealed that GPT failed to produce a single valid SQL query across all three learning rates. Instead of generating the expected SQL outputs, GPT consistently produced natural language responses that closely mirrored the input queries. For example, when presented with the natural language prompt “What is the total payment amount for devices in Texas?”, the model simply paraphrased or repeated the question rather than producing the corresponding SQL query. This behavior persisted across the entire test set.

The evaluation metrics (Table 5) reflect the severity of GPT's failure. Accuracy, precision, and recall were all recorded as zero for each learning rate, as the model did not produce any correct predictions. Interestingly, the F1 score was reported as 1.0000 due to an artifact of the evaluation pipeline. This result arose from the "micro" averaging strategy used during precision-recall calculations, combined with the handling of zero predictions through the zero_division parameter. While this numerical anomaly suggests a non-zero F1 score, it does not imply any actual success, as the GPT model entirely failed to generate valid outputs.

Table 5. Evaluation metrics for fine-tuned GPT.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
GPT finetuned with learning rate 1e-5	0.00	0.00	0.00	1.00

GPT finetuned with learning rate 3e-5	0.00	0.00	0.00	1.00
GPT finetuned with learning rate 5e-5	0.00	0.00	0.00	1.00

These results stand in stark contrast to the performance of T5 and BERT. T5, a sequence-to-sequence model, successfully generated syntactically and semantically correct SQL queries for simpler examples, while BERT, adapted as a classification model, managed to identify SQL query clusters with moderate accuracy. GPT, however, was unable to generalize to the task of SQL query generation under the same conditions. The failure to produce SQL outputs, even for simple examples, underscores the model's inability to adapt to this structured task.

The reasons for GPT's poor performance likely stem from both its autoregressive nature and the inherent complexity of SQL syntax. Unlike T5, which is explicitly designed for sequence-to-sequence tasks, and BERT, which excels in classification tasks, GPT's autoregressive structure is less suited for generating highly structured outputs like SQL queries. GPT's training objective, which focuses on causal language modeling, may not provide sufficient supervision for structured text generation tasks without significantly larger datasets.

In conclusion, while GPT was fine-tuned using the same rigorous methodology and systematic approach applied to T5 and BERT, it completely failed to generate SQL queries from natural language inputs. This outcome highlights a significant shortcoming of GPT's autoregressive architecture in handling structured text generation tasks like SQL query translation, especially in low-data scenarios.

3.7.4 Evaluating the Fine-Tuned Models

To choose the best out of the fine-tuned models tasked with translating natural language queries into SQL, a detailed and systematic evaluation process was implemented. This evaluation was conducted using dedicated notebooks specifically designed to assess the models' performance both syntactically and semantically. The goal was to ensure that the models could generalize effectively to unseen queries, generating SQL statements that were not only syntactically correct but also semantically accurate when executed against a database. Two types of models were evaluated in this process: T5 and BERT, each fine-tuned with variations in hyperparameters, particularly the learning rate.

The evaluation began with the preparation of a dedicated test set consisting of 25 natural language-SQL query pairs. These pairs were excluded from the fine-tuning process to ensure that the evaluation measured the models' ability to generalize to entirely new queries.

The evaluation process for both the T5 and BERT models proceeded in two stages: a string-based comparison and an execution-based validation. In the string-based evaluation, each model's predictions were compared directly to the reference SQL queries using exact string matching. Metrics such as accuracy, precision, recall, and F1 score were calculated based on the number of exact matches between the predicted and reference queries. This approach provided a straightforward assessment of the models' syntactic correctness but also revealed its inherent limitations. String-based comparison fails to account for SQL queries that are semantically equivalent yet differ in syntax.

To address this limitation, the second stage of the evaluation incorporated execution-based validation. In this step, the predicted SQL queries and their corresponding reference queries were executed against the PostgreSQL table, Preprocessed_Test_Data. This ensured that the queries were validated against real data. The results returned by each query were then compared with a small numerical tolerance to account for floating-point differences. This execution-

based approach provided a more robust assessment of the models' performance, ensuring that they were evaluated based on their ability to produce semantically accurate SQL statements.

For the T5 models, three versions fine-tuned with different learning rates —1e-5, 3e-5, and 5e-5— were evaluated using this process. The evaluation notebook loaded each fine-tuned model along with its corresponding tokenizer and generated SQL predictions for each natural language query in the test set. The results (Table 6) showed that the model fine-tuned with a learning rate of 3e-5 outperformed the others. It achieved an accuracy, precision, recall, and F1 score of 44% in the string-based evaluation and a query result match percentage of 64% in the execution-based evaluation. In contrast, the models fine-tuned with learning rates of 1e-5 and 5e-5 achieved lower execution-based match percentages, both at 48%. This result highlighted the importance of execution-based validation in determining the most effective model, as the differences in string-based metrics had been relatively minor during fine-tuning. The superior performance of the 3e-5 model, particularly in execution-based validation, led to its selection as the final solution for the task.

Table 6. Comprehensive evaluation results for fine-tuned T5.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Match Percentage**
T5 finetuned with learning rate 1e-5	0.32	0.32	0.32	0.32	48.0%
T5 finetuned with learning rate 3e-5	0.44	0.44	0.44	0.44	64.0%
T5 finetuned with learning rate 5e-5	0.36	0.36	0.36	0.36	48.0%

For the BERT models, a similar evaluation process was applied using another dedicated notebook. The BERT models were fine-tuned as sequence classifiers, meaning that instead of generating SQL queries directly, they outputted labels corresponding to the most likely SQL query. A pre-defined

label-to-SQL mapping dictionary was used to decode these predicted labels into executable SQL statements. As with the T5 models, the evaluation began with tokenizing each natural language query using the BERT tokenizer and feeding the tokenized inputs into the fine-tuned models. The models' predictions were then evaluated using the same two-stage approach. The results (Table 7) for the BERT models revealed significant challenges in their ability to generate correct SQL queries. Across all three versions, the observed metrics were notably low. Accuracy, precision, recall, and F1 score all remained at 4% in the string-based evaluation, while the execution-based query result match percentage reached only 12%. These findings indicated that the BERT models struggled to generalize effectively to the test queries, frequently producing SQL statements with errors such as incorrect WHERE conditions, mismatched filters, or improperly applied aggregation functions. Despite these difficulties, one BERT model, fine-tuned with a learning rate of 1e-5, demonstrated marginally better performance compared to the others. While the differences were not substantial, this model exhibited slightly greater consistency and reliability during testing, particularly in execution-based validation. Based on these observations, the 1e-5 model was selected as the final BERT solution.

Table 7. Comprehensive evaluation results for the fine-tuned BERT.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Match Percentage**
BERT finetuned with learning rate 1e-5	0.04	0.04	0.04	0.04	12.0%
BERT finetuned with learning rate 3e-5	0.03	0.03	0.03	0.03	8.0%
BERT finetuned with learning rate 5e-5	0.03	0.03	0.03	0.03	8.0%

While both the T5 and BERT models underwent rigorous evaluation, a different decision was made regarding the GPT models. Due to GPT’s exceptionally poor results during the fine-tuning phase, it was deemed unnecessary to proceed with the full evaluation process. The fine-tuning notebooks revealed that GPT

failed to generate any valid SQL predictions. Instead, GPT consistently produced outputs in natural language that closely paraphrased the input prompts rather than structured SQL queries. Given that the evaluation process relied on executing predicted queries against the PostgreSQL table, running GPT predictions was infeasible, as none of its outputs were syntactically valid SQL statements. Consequently, a decision was made to drop GPT entirely from the evaluation phase and the final solution. This was justified, as achieving meaningful performance would have required substantially more training data or additional fine-tuning effort, which fell outside the study's scope and goals.

Overall, the evaluation highlighted the significance of execution-based validation for SQL generation tasks. String-based metrics proved inadequate for assessing semantically correct but syntactically varied outputs. The T5 model fine-tuned at 3e-5 emerged as the best-performing solution, achieving the highest query result match percentage. In comparison, BERT models performed poorly but still produced a slightly better result with the 1e-5 learning rate. Meanwhile, GPT’s exclusion from the evaluation emphasized the importance of architectural suitability and sufficient training when adapting models for structured output tasks like SQL generation.

4 Implementation and Results

The chapter focuses on the implementation of the NLP-driven solution, demonstrating its capability to translate natural language queries into actionable insights within a Business Intelligence (BI) context.

4.1 Building the NLP-Driven Solution Workflow

Although the fine-tuned models were not optimized to achieve their peak performance, the work proceeded with building the final solution. This decision aligned with the study's objectives, as the scope of fine-tuning had been deliberately defined to remain relatively small. The focus shifted toward integrating the fine-tuned models into a functional system that could translate natural language queries into SQL statements, ultimately delivering actionable insights.

Key components of the final solution included a FastAPI-based backend for query processing and generation, a Streamlit-powered user interface for user interaction, a PostgreSQL database for result storage, and Power BI for dynamic visualization. By bringing these technologies together, the system served as a proof-of-concept for how NLP-driven solutions can simplify BI workflows and make data analysis accessible to non-technical users.

The backend, implemented using FastAPI in app.py, served as the central processing unit for query handling. FastAPI's asynchronous capabilities made it efficient for real-time input processing and SQL query generation. The backend exposed an endpoint, /query, which accepted a natural language query along with the user's model selection (T5 or BERT). Upon receiving the request, the input was tokenized and processed by the chosen model to generate a SQL query. This query was executed against the Preprocessed_Dataset table in PostgreSQL, which contained the complete dataset for analysis. SQLAlchemy, a Python ORM, ensured secure query execution using parameterized queries, mitigating the risk of SQL injection. Robust error-handling mechanisms were

implemented through try-except blocks to manage issues such as malformed SQL queries or failed executions. Successfully processed queries, along with their SQL statements and results, were stored in the api_results table, providing a structured record for retrieval and visualization.

The frontend, developed using Streamlit in ui.py, provided an intuitive user interface for query submission. Streamlit's simplicity and lightweight design allowed for rapid implementation and seamless backend integration. Users could enter natural language queries, select the preferred model, and submit their inputs, with the frontend managing API requests via the requests library. Immediate feedback, such as success or error messages, ensured a smooth and transparent user experience.

Once the backend had stored the results in PostgreSQL, Power BI connected to the api_results table to fetch and display the outputs. The results were presented in a dynamic visualization environment, allowing users to view query details and outputs in real time. To ensure the dashboard displayed the latest data without manual intervention, an automated hourly refresh was configured using a Power BI data gateway.

This streamlined system workflow combined FastAPI for efficient backend processing, SQLAlchemy for secure SQL execution, Streamlit for the user interface, and PostgreSQL for data storage. Together, these technologies enabled the seamless transition from natural language inputs to SQL-generated outputs and their visualization.

4.2 The Final NLP-Driven Business Intelligence Solution

The final solution integrates an NLP-driven natural language to SQL query system with a robust Power BI dashboard, providing users with an intuitive way to extract actionable insights from complex datasets. This system seamlessly combines a browser-based user interface, a backend SQL generation process, and dynamic data visualization tools to create a streamlined workflow that bridges natural language processing with practical business intelligence.

The process begins with the user interface, which acts as the entry point for interaction (Figure 2). When the application is running, the UI opens directly in the browser, presenting a simple and accessible interface. Users are prompted to choose between the two fine-tuned models, T5 or BERT, before entering their natural language query or prompt into an input field. Once a query is submitted, the system processes the input in the backend. The selected model generates the corresponding SQL query, which is executed against the Preprocessed_Dataset table. The query, the result, and the model information are then stored in the api_results table within the PostgreSQL database. On the user-facing side, the interface confirms the successful processing of the query and provides a clear message directing the user to check the results in the Power BI dashboard (Figure 3). This seamless guidance ensures that users can easily navigate from input to result visualization.

Figure 2. User interface for query submission (Before input).

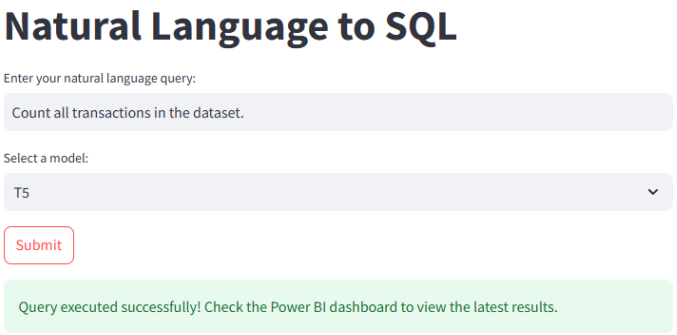

Figure 3. User interface for query submission (After input).

Upon accessing the Power BI dashboard, users can refresh it to fetch the most recent query results from the api_results table. The dashboard's first section focuses on dynamically presenting the outputs generated by the natural language SQL solution (Figure 4). A table visual displays the details of each query, including the natural language question, the selected model, the SQL query generated by the model, and the resulting output (Figure 5). To allow flexible exploration, two slicers are integrated into this table. The first slicer enables filtering the queries based on the selected model, allowing users to display results generated by T5, BERT, or both (Figure 6). The second slicer provides a time filter, enabling the user to refine the view based on specific time ranges (Figure 7). In addition to the table, a card visualization dynamically highlights the result of the most recent query, ensuring immediate visibility and clarity for the user (Figure 8). By combining the table and card visuals, this section allows users to quickly validate their query outputs and interactively explore the results generated by the system.

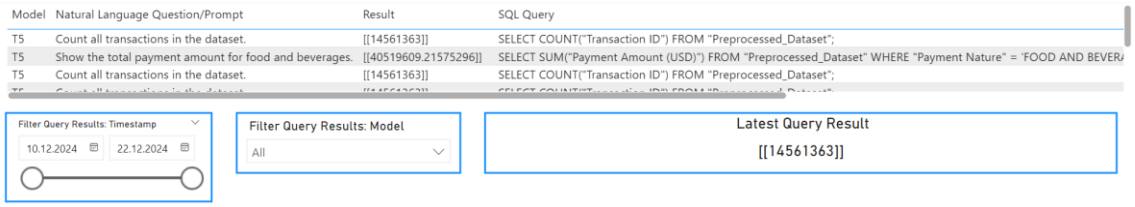

Figure 4. NLP-generated query results in the Power BI dashboard.

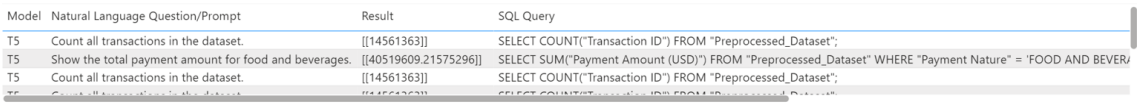

Figure 5. Query details in a table visualization.

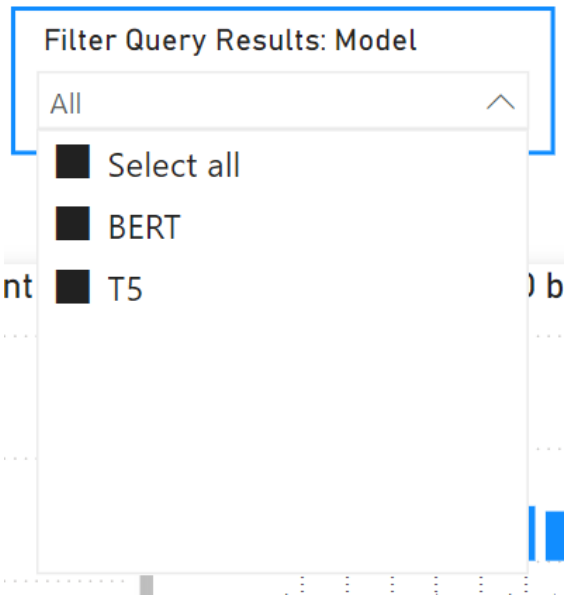

Figure 6. Slicer for model-based filtering of queries.

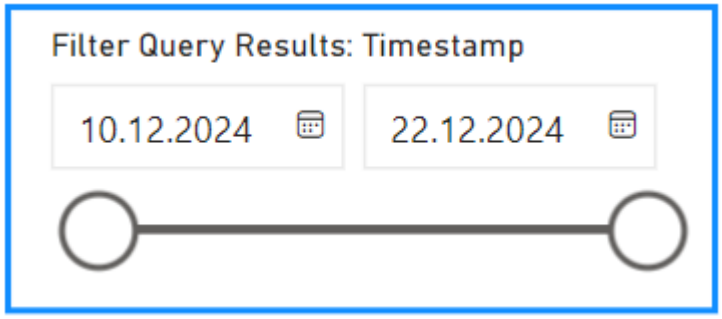

Figure 7. Slicer for time-range filtering of queries.

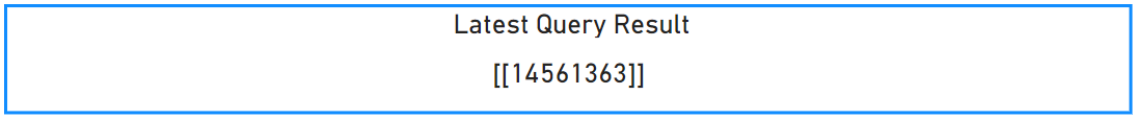

Figure 8. Dynamic display of most recent query result.

The second section of the dashboard complements the natural language SQL solution by offering traditional business intelligence analysis (Figure 9). Unlike the first section, this part of the dashboard provides static insights derived from the Preprocessed_Dataset table. Designed for users who prefer predefined analyses, the section includes a series of interactive visuals to explore trends

and patterns in the data. Two clustered column charts provide insights into recipients and manufacturers by aggregating transaction counts based on recipient types and manufacturer names. A stacked column chart breaks down transaction counts across product categories, distinguishing between payment natures such as drugs or devices. A map visualization highlights geographic trends by plotting states and sizing points based on the distinct count of recipient profile IDs, enabling users to identify areas with the highest or lowest transaction volumes. For temporal analysis, a line chart displays transaction counts over time by plotting payment months, while an area chart highlights total payment amounts for each month. Together, these visuals offer a comprehensive and static overview of the dataset, allowing users to explore trends related to recipients, manufacturers, products, geographic regions, and time-based activity.

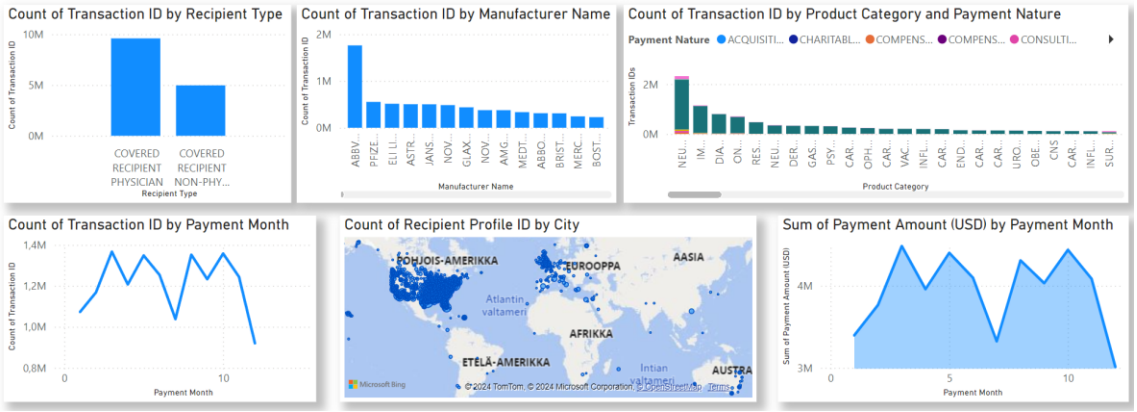

Figure 9. Power BI dashboard section showcasing BI insights.

This dual functionality of the Power BI dashboard demonstrates the solution's versatility. The first section caters to users who want specific insights without the need for technical expertise, allowing them to interact with the system using natural language. By generating and executing SQL queries dynamically, the system ensures that even non-technical users can extract meaningful results. The second section, with its traditional business intelligence visuals, serves as a complementary layer for broader analysis, offering predefined insights that users can rely on for a static, high-level view of the dataset.

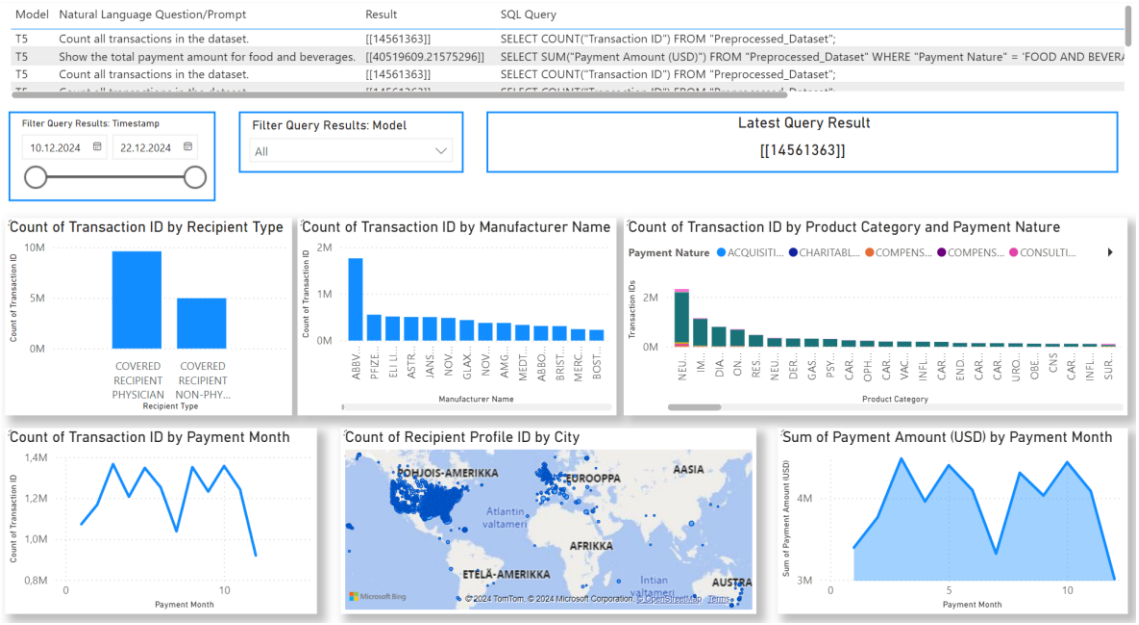

Figure 10. Comprehensive view of the NLP-driven business intelligence dashboard.

The entire system workflow, from query input to result visualization, is both intuitive and efficient. By combining natural language input with robust visualization tools, the system bridges the gap between advanced AI capabilities and real-world data analysis. Users can extract insights, validate outputs, and make informed business decisions without the need for extensive technical expertise. The solution highlights how NLP-driven systems can complement traditional BI tools, providing a scalable and accessible approach to enhancing data-driven decision-making.

4.3 Results and Discussion

This section presents the key findings from the implementation of the NLP-driven BI system and reflects on its usability, performance, and potential challenges. The discussion also considers the system's accessibility for non-technical users and identifies areas for future improvements.

4.3.1 Analysis of System Performance

The implementation of the NLP-driven solution showcased several strengths. The integration of fine-tuned T5 and BERT models into a functional system, complete with a FastAPI backend and a Streamlit-based user interface, demonstrated how Natural Language Processing (NLP) models can operationalize natural language-to-SQL query translation. The system's ability to transform user-friendly natural language inputs into actionable SQL queries and subsequently visualize the results in Power BI highlights its practical utility.

Model performance varied: T5 outperformed BERT in generating syntactically correct SQL queries for a wider range of inputs. However, both models struggled with complex or ambiguous queries, reflecting the inherent challenges in capturing the nuanced semantics of natural language inputs. The integration of the solution components was also a noteworthy success. FastAPI ensured efficient backend processing, while Power BI's dashboard offered a dynamic environment for presenting query outputs. Together, these elements bridged the gap between technical AI models and accessible user interfaces.

Additionally, the integration of NLP into BI demonstrated strong potential as a user-friendly and visually appealing solution. Hypothetically, the system would offer an accessible way for non-technical users to interact with BI tools, highlighting the effectiveness of combining NLP with BI to simplify data analysis processes. This finding aligns with the literature's emphasis on enhancing BI accessibility through advanced technologies.

4.3.2 Hypothetical User Experience

Although no user testing was conducted, the design and functionality of the system suggest it would be well-received by non-technical users. The intuitive user interface, powered by Streamlit, simplifies the process of inputting natural language queries and selecting preferred models. This aligns with the goal of making Business Intelligence (BI) tools more accessible. The Power BI

dashboard further enhances usability by offering a clear and organized presentation of results, allowing users to interactively explore data without requiring technical expertise.

From a hypothetical standpoint, users would likely appreciate the system's ability to dynamically generate and execute queries based on their inputs. However, some frustration could arise from the limitations in query handling, particularly for more complex or ambiguous requests. Addressing these shortcomings in future iterations, such as through more advanced fine-tuning, larger datasets, or additional user guidance, would further enhance user satisfaction and trust in the system.

4.3.3 Challenges and Future Improvements

While the NLP-driven solution successfully demonstrated the feasibility of integrating natural language processing into Business Intelligence, several challenges emerged, particularly in terms of scalability, query accuracy, and usability evaluation.

A critical factor in the system's success was the preparatory processes, including dataset preprocessing and model fine-tuning. These steps played a pivotal role in ensuring that the models could accurately interpret and translate natural language inputs into SQL queries. However, due to the deliberately limited scope of this study, scalability and real-world user testing were not prioritized, making them key areas for future development.

One of the most significant challenges is scalability. The current implementation operates effectively on a controlled dataset but expanding it to support larger datasets and more diverse BI use cases would require substantial improvements. Future iterations should consider optimizing model performance for more complex query structures, increasing dataset diversity, and incorporating adaptive learning techniques to improve accuracy over time.

Another major limitation is the lack of real-world user testing. Since the system was developed as a proof of concept, its usability was assessed only in a theoretical context. While the interface is designed to be intuitive, practical testing with end users would provide essential feedback on its effectiveness, accessibility, and potential areas of confusion. Conducting usability studies would allow for iterative design improvements, ensuring that the system meets the needs of non-technical users in real business environments.

Query accuracy also remains a challenge, particularly when handling complex or ambiguous natural language inputs. While the models performed well for straightforward queries, more intricate or vague user inputs sometimes led to incorrect SQL translations. Future improvements could involve further fine-tuning with more diverse training data, the integration of domain-specific models, or the use of hybrid approaches that combine NLP with rule-based query refinement to enhance precision.

Additionally, advancements in NLP techniques could further improve system performance. Refining the model through additional fine-tuning on more specialized SQL query datasets could improve adaptability and accuracy. Expanding the dataset to include a broader variety of SQL query patterns would also help improve model generalization and reduce errors in real-world applications.

Finally, while this study focused on feasibility rather than deployment, future research should explore how such systems could be integrated into enterprise-level BI solutions. This would require evaluating factors such as security, system reliability, and computational efficiency to ensure scalability beyond small-scale applications.

5 Conclusion and Discussion

This thesis set out to explore how Natural Language Processing (NLP) can be integrated into Business Intelligence (BI) workflows to improve accessibility and decision-making. The objective was to develop and evaluate an NLP-driven system capable of translating natural language queries into SQL statements, enabling non-technical users to interact more intuitively with BI tools. Through the implementation of a prototype, this thesis demonstrated both the potential and limitations of such a system. While the developed solution successfully processes user queries and provides visualized insights, challenges related to scalability, model fine-tuning, and real-world evaluation remain.

This chapter summarizes the key findings of the thesis, discusses its limitations, explores its business implications, and provides recommendations for future research and development.

5.1 Summary of Findings

This thesis demonstrates the feasibility of integrating NLP models into BI workflows by enabling natural language interaction with structured data. The system developed serves as a proof of concept, showcasing how NLP-driven query translation can improve data accessibility. The results indicate that NLP-powered BI solutions could enhance usability, allowing users to retrieve insights without requiring SQL expertise. The integration of an NLP-driven query interface with Power BI visualization demonstrated how interactive dashboards can further improve data accessibility by providing users with an intuitive way to explore results.

However, despite its effectiveness in handling structured queries, the system faces challenges when processing complex or ambiguous inputs. The findings align with prior studies emphasizing that Self-Service BI (SSBI) tools require more advanced NLP integration to ensure accurate interpretation of natural language queries (Al-Okaily et al., 2023). Additionally, this thesis supports

research indicating that transformer-based models, such as BERT and T5, can improve natural language query processing within BI applications (Wang et al., 2023).

A further challenge identified is query accuracy and model performance variation. The results showed that T5 generated syntactically correct SQL queries more consistently than BERT, particularly for complex inputs. However, both models struggled with multi-step queries and ambiguous phrasing, reinforcing prior findings that fine-tuning on domain-specific data is essential for improving accuracy in NLP-driven BI solutions (Kotei & Thirunavukarasu, 2023).

Additionally, GPT completely failed at the task. Despite undergoing the same fine-tuning process as T5 and BERT, it did not generate a single valid SQL query. Instead, it consistently repeated or paraphrased the natural language input rather than producing structured SQL statements. This result aligns with research suggesting that GPT's autoregressive nature makes it ill-suited for structured text generation tasks like SQL translation without extensive fine-tuning and large-scale training data (Wang et al., 2023).

Additionally, scalability remains a limitation. While the system processed queries efficiently in a controlled setting, enterprise-level deployment would require integration with larger-scale data architectures. Previous research suggests that BI scalability can be improved by leveraging data warehouses and cloud-based infrastructures (Nambiar & Mundra, 2022).

Despite these challenges, the findings confirm that NLP-driven BI solutions hold significant potential. The system successfully demonstrated how NLP models, combined with interactive dashboards, can simplify BI workflows, making data insights more accessible. However, further research is needed to improve model accuracy, scalability, and real-world usability testing before such systems can be widely adopted.

5.2 Limitations and Constraints

Despite its contributions, this thesis faced several limitations. One major constraint was the lack of real-world user testing, which restricted the ability to assess the system's usability in practical settings. While the design aimed to support non-technical users, empirical feedback from business professionals would provide deeper insights into its effectiveness. Prior research highlights that user engagement and iterative feedback loops are critical for refining NLP-powered BI solutions (Passlick et al., 2023).

Scalability remains a challenge, as the system was tested on a controlled dataset. While it processes queries efficiently in this environment, its performance with large-scale enterprise data is uncertain. Research suggests that leveraging data warehouses and cloud infrastructures could improve the scalability and responsiveness of NLP-driven BI systems (Nambiar & Mundra, 2022).

Model accuracy also presented limitations, particularly for complex, ambiguous, or multi-step queries. While T5 and BERT generated correct SQL outputs for simple cases, performance declined with more intricate queries, reinforcing the need for domain-specific fine-tuning (Kotei & Thirunavukarasu, 2023). GPT, despite undergoing the same fine-tuning process, failed to generate valid SQL queries and consistently paraphrased the input instead. This outcome aligns with findings that GPT's autoregressive structure makes it ill-suited for structured query generation without extensive fine-tuning (Wang et al., 2023).

Ethical considerations, such as bias mitigation and data transparency, must also be addressed as these systems evolve. NLP models can reflect biases present in their training data, leading to unintended skewed outputs. Ensuring fairness, explainability, and regulatory compliance in NLP-driven BI applications will be essential for their adoption (Siau & Chen, 2020).

5.3 Implications for Business and Practical Applications

The findings of this thesis suggest that NLP-driven BI solutions can improve accessibility, efficiency, and decision-making by enabling non-technical users to interact with data using natural language. This reduces reliance on technical analysts and supports a more data-driven business culture.

Natural language interfaces make complex data retrieval tasks significantly more intuitive, allowing decision-makers to extract insights without SQL expertise. This aligns with research indicating that Self-Service BI (SSBI) tools enhance accessibility and efficiency (Al-Okaily et al., 2023). The integration of an NLP-driven query interface with Power BI demonstrated how automated query execution and visualization can streamline data analysis and improve real-time decision-making.

Despite these benefits, businesses must consider ethical and security risks. NLP models can introduce biases and generate misleading outputs, impacting decision-making. Ensuring transparency, bias detection, and ethical AI governance will be crucial for maintaining trust in NLP-powered BI systems (Siau & Chen, 2020).

5.4 Future Directions and Recommendations

To advance NLP-powered BI systems, future work should focus on improving scalability, accuracy, and usability. Integrating these solutions with cloud-based infrastructures and large-scale data warehouses could enhance their ability to process complex queries efficiently (Mei et al., 2024).

Model accuracy could be further refined through domain-specific fine-tuning. Training models on specialized SQL datasets could improve precision and adaptability (Kotei & Thirunavukarasu, 2023).

Real-world usability testing is another critical area for future research. While this thesis evaluated the system in a controlled environment, direct testing with

business users would provide insights into usability challenges and improve adoption (Passlick et al., 2023).

Addressing ethical and security concerns will also be necessary. Bias mitigation, data transparency, and explainable AI methods must be incorporated to ensure NLP-powered BI tools align with regulatory requirements and maintain user trust (Siau & Chen, 2020).

In summary, while this thesis provides an initial exploration of NLP-powered BI solutions, further research is needed to refine model accuracy, scalability, and usability. Addressing these challenges will be essential for ensuring the practical deployment of such systems in real-world business environments. By continuing to develop more adaptable, transparent, and robust NLP-driven BI tools, future advancements can enhance the accessibility and effectiveness of data-driven decision-making.

References

Al-Okaily, A., Teoh, A. P., & Al-Okaily, M. (2023). Evaluation of data analytics-oriented business intelligence technology effectiveness: An enterprise-level analysis. Business Process Management Journal, 29(3), 777-800. https://doi.org/10.1108/BPMJ-10-2022-0546

Adewusi, A. O., Okoli, U. I., Adaga, E., Olorunsogo, T., Asuzu, O. F., & Daraojimba, D. O. (2024). Business intelligence in the era of big data: A review of analytical tools and competitive advantage. Computer Science & IT Research Journal, 5(2), 415-431. https://doi.org/10.51594/csitrj.v5i2.791

Dask. (2024). Dask documentation: Parallel and distributed computing. Accessed 21 December 2024. https://docs.dask.org/en/stable/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv. https://doi.org/10.48550/arXiv.1810.04805

Just, J. (2024). Natural language processing for innovation search – Reviewing an emerging non-human innovation intermediary. Technovation, 129, 102883. https://doi.org/10.1016/j.technovation.2023.102883

Kotei, E., & Thirunavukarasu, R. (2023). A systematic review of transformer-based pre-trained language models through self-supervised learning. Information, 14(3), 187. https://doi.org/10.3390/info14030187

Mei, T., Zi, Y., Cheng, X., Gao, Z., Wang, Q., & Yang, H. (2024). Efficiency optimization of large-scale language models based on deep learning in natural language processing tasks. arXiv. https://doi.org/10.48550/arXiv.2405.11704

Nambiar, A., & Mundra, D. (2022). An overview of data warehouse and data lake in modern enterprise data management. Big Data and Cognitive Computing, 6(4), 132. https://doi.org/10.3390/bdcc6040132

OpenAI. (2024). ChatGPT (December 21 version). Accessed 2 December 2024. https://chat.openai.com/

Paaß, G., & Giesselbach, S. (2023). Foundation models for natural language processing: Pre-trained language models integrating media. Springer International Publishing. https://doi.org/10.1007/978-3-031-23190-2

Paradza, D., & Daramola, O. (2021). Business intelligence and business value in organisations: A systematic literature review. Sustainability, 13(20), 11382. https://doi.org/10.3390/su132011382

Passlick, J., Grützner, L., Schulz, M., & Breitner, M. H. (2023). Self-service business intelligence and analytics application scenarios: A taxonomy for differentiation. Information Systems and e-Business Management, 21, 159-191. https://doi.org/10.1007/s10257-022-00574-3

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. OpenAI. Retrieved from https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1-67. https://doi.org/10.48550/arXiv.1910.10683

Siau, K., & Chen, X. (2020). Business analytics/business intelligence and IT infrastructure: Impact on organizational agility. Journal of Organizational and End User Computing, 32(4), 138-161. https://doi.org/10.4018/JOEUC.2020100107

Tavera Romero, C. A., Ortiz, J. H., Khalaf, O. I., & Ríos Prado, A. (2021). Business intelligence: Business evolution after industry 4.0. Sustainability, 13(18), 10026. https://doi.org/10.3390/su131810026

U.S. Centers for Medicare & Medicaid Services. Open Payments Data. Accessed on 14 September 2024. https://openpaymentsdata.cms.gov/

Wang, H., Li, J., Wu, H., Hovy, E., & Sun, Y. (2023). Pre-trained language models and their applications. Engineering, 25, 51-65. https://doi.org/10.1016/j.eng.2022.04.024